GDB Commands — Quick Reference & Examples

This document summarizes common GDB commands with a short example session and a minimal C program you can use to try them out.

1. Preparation

Compile your program with debugging symbols and without optimization:

```
gcc -g -O0 example.c -o example
```

2. Minimal example program

Save the following as example.c:

```
#include <stdio.h>

int factorial(int n) {
    if (n <= 1) return 1;
    return n * factorial(n - 1);
}

int main(void) {
    int r = factorial(5);
    printf("result = %d\n", r);
    return 0;
}
```

3. Common GDB commands (summary)

Run / control execution

```
run (r)                -- start the program (optionally with args)
continue (c)            -- continue after stop
kill                    -- kill the inferior process
quit (q)                -- exit gdb
```

Breakpoints

```
break <func>            -- break at function entry (e.g. break
factorial)
break <file>:<line>     -- break at source line (e.g. break
example.c:12)
tbreak <loc>            -- temporary one-shot breakpoint
delete <num>            -- remove breakpoint by number
info breakpoints        -- list breakpoints
```

Stepping & navigation

```
step (s)                -- step into (source-level)
next (n)                -- step over (source-level)
```

```
finish                   -- run until current function returns
continue (c)             -- resume execution
```

Stack & frames

```
backtrace (bt)          -- show call stack
frame <n>                -- select frame number n
up / down                -- move up/down the call stack
info args                -- show function arguments in current frame
info locals              -- show local variables
```

Inspect / modify

```
print <expr>            -- print expression or variable (e.g. print i)
set var <expr>          -- change variable value (e.g. set var i = 10)
display <expr>          -- automatically display expr when program stops
watch <expr>            -- stop when expr is written
rwatch <expr>           -- stop when expr is read
awatch <expr>           -- stop on read/write
```

Files & source

```
list                    -- list source around current line
list <file>:<line>      -- list around given file:line
info source             -- info about current source file
```

4. Example GDB session (step-by-step)

Commands to run after compiling the example program above:

```
gdb ./example
(gdb) break factorial
(gdb) run
# program stops at first call to factorial
(gdb) bt
# see stack of recursive calls
(gdb) frame 3
(gdb) info args
(gdb) info locals
(gdb) print n
(gdb) continue
(gdb) break factorial if n==1
(gdb) run
# program will stop only at the base-case call where n==1
```

5. Debugging a crash (segfault) example

If your program crashes with SIGSEGV, useful commands are:

```
run
# after crash:
(gdb) bt
(gdb) frame <n>
```

```
(gdb) info locals
(gdb) print <pointer>
(gdb) x/32bx <address>   # examine memory bytes
```

6. Tips & useful options

- Use '-fno-omit-frame-pointer' when compiling to improve backtraces on optimized builds.

- Use 'set pagination off' to avoid --More-- prompts in scripts.

- Use 'set backtrace limit <n>' to limit huge recursive traces.

- Combine with AddressSanitizer (gcc -fsanitize=address) to catch memory bugs reliably.

7. GDB command-file (automation)

You can put GDB commands in a file (gdbcmds.txt) and run with 'gdb -x gdbcmds.txt ./example'. Example content:

```
break factorial
run
bt
info locals
quit
```

8. Quick reference table

```
run, r           -- start the program
break, b         -- set breakpoint
tbreak           -- temporary breakpoint
continue, c      -- resume
step, s          -- step into
next, n          -- step over
finish           -- run until return
backtrace, bt    -- show call stack
frame <n>        -- switch to frame n
print, p         -- print expression
watch            -- watchpoint (stop on write)
info locals      -- show locals
info args        -- show args
```

9. Example: Finding a bug (segfault) quickly

A short workflow:

1. Compile with -g -O0. 2. Run inside gdb. 3. On crash, 'bt' to see where. 4. Inspect locals and pointers. 5. Set breakpoints / watchpoints to reproduce the bad write.